**Clustering**

**1. DBSCAN**

Using DBSCAN iterate (for-loop) through different values of min\_samples (1 to 10) and epsilon (.05 to .5, in steps of .01) to find clusters in the road-data used in the Lesson and calculate the Silohouette Coeff for min\_samples and epsilon. Plot one line plot with the multiple lines generated from the min\_samples and epsilon values. Use a 2D array to store the SilCoeff values, one dimension represents min\_samples, the other represents epsilon.

Expecting a plot of epsilon vs sil\_score.

all\_scores = []

for min\_sample in min\_samples:

scores = []

for epsilon in epsilons:

# calculate silouette score here

score =

scores.append(score)

all\_scores.append(scores)

**2. Clustering your own data**

Using your own data, find relevant clusters/groups within your data (repeat the above). If your data is labeled with a class that you are attempting to predict, be sure to not use it in training and clustering.

You may use the labels to compare with predictions to show how well the clustering performed using one of the clustering metrics (http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation).

If you don't have labels, use the silhouette coefficient to show performance. Find the optimal fit for your data but you don't need to be as exhaustive as above.

Additionally, show the clusters in 2D or 3D plots.

As a bonus, try using PCA first to condense your data from N columns to less than N.

Two items are expected:

* Metric Evaluation Plot (like in 1.)
* Plots of the clustered data